LA-UR-22-21250

Approved for public release; distribution is unlimited.

Title:	Research software engineering: Professionalization, roles, and identity
Author(s):	Sims, Benjamin Hayden
Intended for:	Report
Issued:	2022-02-14

Research software engineering: Professionalization, roles, and identity

Benjamin Sims
Statistical Sciences Group (CCS-6)
Los Alamos National Laboratory

February 8, 2021

Table of Contents

Abstract **2**

Introduction **2**

Data and methods **3**

Roles and identities of research software engineers **5**

Research software engineer and other professional identities 5

Models of professional identity 7

Dominance models 8

Multiplicity models 9

Summary 11

Research software professionals at national laboratories **11**

Professionalization in software **13**

The sociology of professions 13

History of the software engineering profession 14

Research software engineering as a profession 16

Conclusion **17**

References **19**

Abstract

This report summarizes the results of a study of research software professionals based on 17 interviews conducted at U.S. national laboratories and academic institutions. The study focuses on understanding the emerging professional role of research software engineer (RSE) and examining how the history of software engineering might be relevant to the emergence of research software engineering as a professional movement. The report first uses interview data to identify several models of professional identity for research software professionals. These include dominance models in which a person strongly identifies with a single professional identity (such as RSE), and multiplicity models in which an individual sees an identity like RSE as just one facet of their overall professional role. Next, it identifies three types of work situations for software professionals at national laboratories, covering those who work in dedicated RSE organizations, those who play an integral role in large multiphysics code projects, and those who work in a more isolated capacity to provide software development support for less computationally intensive research projects. It then reviews some key ideas from the sociology of professions in the context of the software engineering profession, suggesting parallels between the current RSE professional movement and aspects of the history of software engineering. It concludes by suggesting some potential practical implications of these findings for the RSE movement and professional organizations.

Introduction

Research software engineer (RSE) is an emerging professional role within the global scientific computing community. There are currently RSE professional organizations in many countries, including the US-RSE association in the United States. The general goal of the RSE movement appears to be to provide a positive professional identity and clearly defined professional role, or in some cases a job title, for people who work primarily as software developers in research settings. In general, the movement appears to have emerged in relation to two developments in the research computing world: First, research software codes are becoming increasingly complex and difficult to manage, leading to increasing reliance on individuals who have strong software development skills, and creating a situation where some research team members may spend all of their time on software development and maintenance. Second, these individuals, many of whom have PhDs in scientific fields, appear to have become increasingly dissatisfied with how they are treated in these roles, feeling that they are undervalued and marginalized within the status hierarchies of research institutions. These developments may have been the impetus that encouraged some of these professionals to start working together to develop the concept of the research software engineer.

As an emerging professional identity, the RSE role is still imprecisely defined (often intentionally so) and its professional institutions are still in the early stages of their development. As such, there are many possible directions the RSE movement may take in the coming years, and definitional issues around professional identity and boundaries are still under active discussion. This report aims to describe and provide some useful definitions around these issues that will

help RSEs, leaders of RSE organizations, and others who support this professional movement plan for its future development.

This study was funded as part of the IDEAS-Classic project (https://ideas-productivity.org/ideas-classic), a partnership between the Advanced Scientific Computing Research (ASCR) and Biological and Environmental Research (BER) offices within the Department of Energy (DOE) Office of Science. IDEAS is an acronym for Interoperable Design of Extreme-scale Application Software, and encompasses a number of initiatives related to improving scientific software quality, productivity, and teams.[1] This report contributes to this effort by providing a deeper look at key aspects of the emerging RSE role, which has the potential to be highly relevant for extreme-scale scientific software development projects. Because of its limited scope, this study focuses on identifying a few key questions about the roles and identities of RSEs and making some initial connections to related research literature. There are many important aspects of the RSE role and professional movement that are not addressed here, but would likely be fruitful subjects for future research. For example, a more comprehensive study of the origins of RSE organizations in different countries could help clarify where the movement originated, how the original members came to conceive of themselves as occupying a common professional niche, and how institutional stakeholders have been persuaded to buy into this concept. Ethnographic or interview-based studies of RSEs at different institutions could provide more clarity on the roles and practices of RSEs in relation to other participants in scientific computing work. These kinds of studies could also provide more opportunities to connect the RSE phenomenon to historical and social science research on science, professions and computing.

The remainder of this report is divided into five sections. The first section describes data and methods used in this study. The next three sections provide distinct perspectives on the emergence of RSE as a professional role. The first examines the professional identities and work situations of RSEs and similar research software professionals, in order to understand what motivates people to identify as an RSE. The second looks at how this role resonates (or fails to resonate) in different work settings, specifically at U.S. national laboratories. The third provides a brief survey of the sociology of professions, with a specific focus on software engineering, providing some insights relevant to the future evolution of the RSE role. Finally, the conclusion discusses the implications of these findings for how RSE professional organizations might most effectively structure themselves and reach out to the research software community, and how national laboratories may better serve research software professionals within their organizations.

Data and methods

This study relies on three main sources of information:

[1] IDEAS-Classic led to a number of other initiatives related to improving scientific software quality, productivity, and teams, including IDEAS-Watersheds, which focuses on hydrological codes; IDEAS-ECP, part of DOE's exascale computing project (ECP); and xSDK4ECP, an extreme-scale software development kit.

1. *Interviews:* The author conducted 17 interviews with research software professionals via remote video conferencing. All of the subjects were based in the US, 3 at academic institutions and 14 at national laboratories. Interviews were captured in detailed notes, and were also recorded, except for three interviews where the participant opted out of being recorded. Participants were selected on a somewhat ad-hoc basis, based on referrals from colleagues within the research software community. All of them met the very broad definition of an RSE provided by the US-RSE association, namely "those who regularly use expertise in programming to advance research." However, the author deliberately sought out interviewees who did not identify as RSEs, as well as those who did, in order to capture the views of those who might be skeptical or resistant to associating with the RSE role as well as those who embrace the term. Interviewees participated with the assurance their identities would be protected, and the project was conducted with the approval of the Central Department of Energy Institutional Review Board.

2. *Primary sources:* The author collected papers, presentations, website content, and other resources associated with the RSE movement, including those produced by the US-RSE association and its counterparts in other countries. One important source of information was Vanessa Sochat's *RSE Stories* podcast series (http://us-rse.org/rse-stories), in which she and her collaborators have interviewed dozens of RSEs covering a wide gamut of research fields and institutions. This report draws on these sources only as background information, as a systematic textual analysis was not possible within the scope of the project.

3. *Sociological and historical studies of computing professions:* These studies were used to situate the RSE movement in relation to larger historical trends in the development of the computing and software professions.

The author captured detailed notes electronically during interviews, then coded and analyzed them for thematic content using QDA Miner qualitative data analysis software (https://provalisresearch.com/qdaminer). This analysis was used to identify particularly significant interview segments, which were then transcribed directly from interview recordings. The interview excerpts in this report are drawn from these transcribed segments, except for a few short quotations that were reconstructed from notes where a recording was not available. Interviews were also categorized by type of institution (university or national laboratory) and professional identity or identities reported by the interviewee, which enabled some basic quantitative analysis.

This is a small-scale, primarily qualitative study, and participants were recruited by informal networking within U.S. institutions. As such, the findings may be incomplete and/or biased by the recruitment method. This approach is appropriate for gaining an initial understanding of potential motivations and professional identities of research software professionals, but in

order to provide a more definitive picture, it would ideally be followed up by further or larger studies using a variety of methods.

Roles and identities of research software engineers

Research software engineer and other professional identities

Interviewees were selected for this study based on having job duties that appeared RSE-related, not based on whether they viewed themselves as RSEs. To address issues of professional identity, one of the interview questions asked whether participants considered themselves to be an RSE (only), an RSE and something else, or some other professional identity.

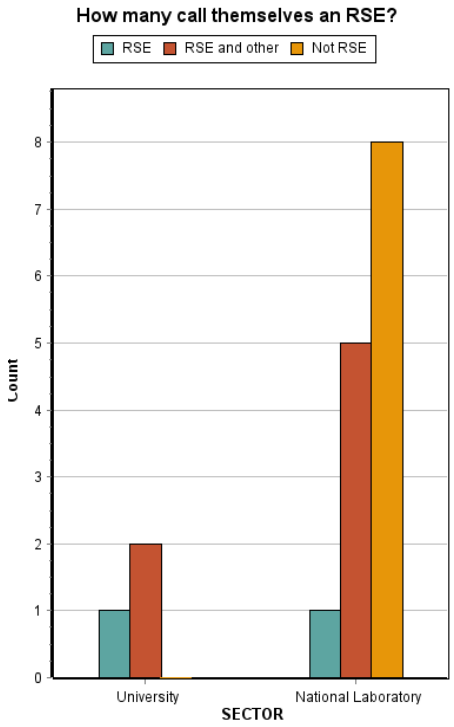

Figure 1. Professional identities reported by interviewees, by sector.

As shown in Figure 1, only 2 interviewees identified themselves as an RSE exclusively, while an additional 7 identified themselves as an RSE and something else. 8 respondents did not identify themselves as RSEs at all, even though 5 of the 8 were familiar with the term. Although this is not a representative or statistically relevant sample, these numbers do suggest that RSE might not yet be a particularly strong professional identity, even for those who do call themselves RSEs.

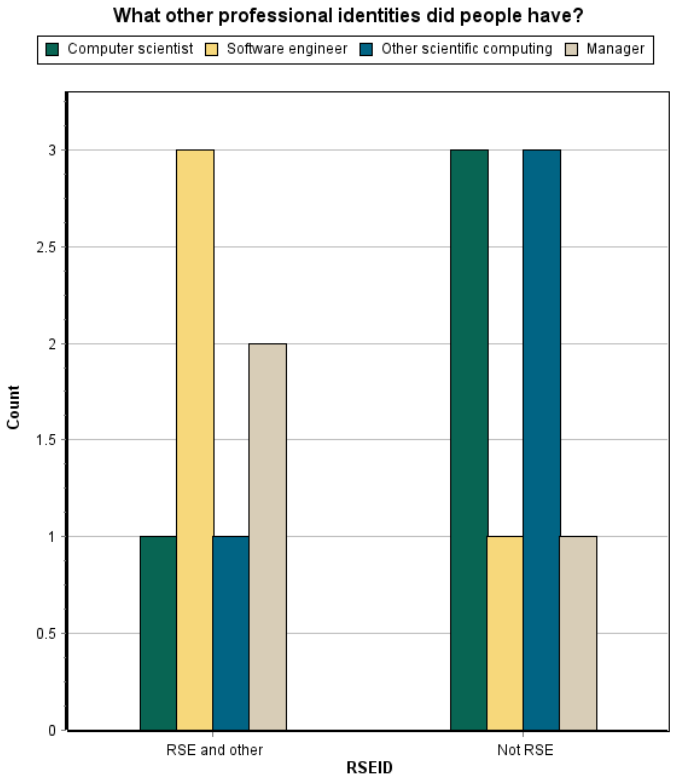

Figure 2. Non-RSE professional identities reported by interviewees. Left side shows the primary non-RSE professional identities of interviewees who identified as RSE and another identity; right side shows primary professional identities of those who did not identify as RSEs.

Another way of looking at RSE identity is through how it relates to other professional identities in scientific computing. As indicated in Figure 2, the most common identities mentioned by interviewees, other than RSE, were software engineer, computer scientist, and manager. Other identities are grouped under the heading of "other scientific computing," which mainly covers a variety of specialized roles in high performance computing (HPC), such as people who optimize codes to run on specialized hardware, or who primarily identify as domain scientists but spend most of their time on computing. The left side of Figure 2 shows that, among those who identified as RSEs and some other professional role, the most common other roles were software engineer and manager (the latter mainly covered RSEs who managed other RSEs). This suggests that people may see software engineer and manager roles as more similar to the RSE role, while computer scientist and other scientific computing roles are not quite as closely tied. The right side of figure 2 shows that people who did not identify as RSEs were most likely to identify as computer scientists or another scientific computing role (again, mainly in HPC). This again suggests that these roles may be seen as more distinct from the RSE role. As discussed below, this may reflect the very particular culture around HPC at some national labs, where status hierarchies mean that most people who have PhDs prefer to identify as scientists rather than engineers, and the complexity of HPC machines and their programming has spawned a

range of specialized computing roles that do not map neatly onto the research software engineering paradigm.

Models of professional identity

In the sociological literature, professional identity refers to an individual's sense of self-identification with a profession or role. Recent research in this area has shown that professional identities are often complex, and this complexity may be increasing over time as boundaries of newer professions may be less clear and people are increasingly asked to play hybrid professional roles. For example, Caza and Creary (2016) identify five models of professional identity: *Dominance*, where a person defines themselves in terms of a single primary profession, even if they sometimes do other types of work; *intersection*, where a person sees themselves as occupying an intersection between two professional roles, e.g. nurse-midwife; *compartmentalization,* where a person defines their work in terms of more than one profession, but sees themselves as occupying different professional roles at different times; *holism,* where a person defines a unique professional role for themselves that encompasses aspects of multiple professions; and *augmentation*, where an individual sees themselves as occupying multiple, complementary professional roles at the same time. These models are summarized in Table 1.

Professional identity model	**Description**	**Example**
Dominance	Single primary professional role	Software developer
Intersection	Occupying only the overlap between two professional roles	Computational physicist
Compartmentalization	Different professional roles at different times	Sometimes a physicist, sometimes a software developer
Holism	Unique role encompassing aspects of multiple professions	"I guess you could call me an algorithm wrangler"
Augmentation	Multiple professional roles at the same time	Physicist plus software developer plus manager

Table 1. Professional identity models from Caza and Creary (2016). Hypothetical research software-related examples by the author.

Interview subjects reported a number of reasons why they identified as an RSE or not, or why they identified as an RSE in some circumstances but not in others. These responses indicate that RSEs and other scientific computing professionals describe their identities in two main ways, which connect to the models listed above. One commonly expressed model was *dominance*, in which an interviewee described themselves as just an RSE, or just some other professional role. These interviewees often saw themselves as strongly affiliated with a particular professional

role, even if that role had unclear boundaries or they did a variety of related work. In addition, they appeared to see their professional role as an important element of their personal identity, or the identity they wanted to project to others. Another model emphasized *multiplicity* of professional identities and drew sharper boundaries between identities in terms of the tasks they involved, similar to the compartmentalization and augmentation models described above. Interviewees following multiplicity models appeared to see their professional role less as an element of personal identity, and more as a set of tasks or areas of expertise they applied in different work situations.

Dominance models

Interviewees mentioned several variations on the dominance model. One variation came from those who had a strong sense that the RSE role exactly described and provided a legitimate name for the work they had already been doing. One interviewee spoke about this as a way of resolving a kind of imposter syndrome:

> When you're in that academic environment before there was US-RSE, and you're a grad student, and you realize that you love writing software more than you like the research, you feel like an imposter … if I were in grad school right now, and I felt like that, and I heard [about the RSE role] I'd [think] there's a place for me, these are my people, it is okay that I am the way that I am.

As this quote indicates, some who identify as RSEs, particularly in academic settings, previously felt marginalized in the scientific enterprise, without a distinct or valued role, which in turn led them to doubt their own career choices. The role of RSE may resonate strongly with these people specifically because it allows them to articulate a positive identity and a sense of group belonging. Others coming from this perspective more strongly emphasized the value of the RSE title as a way of shaping others' perceptions of their role, and providing a distinct career path:

> In my opinion, the biggest thing is having a bucket to fit in. Organizationally … you have a bucket for people who are domain science researchers. You have a bucket for people who are IT support. You have a bucket … for people who are doing computer science research. And … the career paths and the organization assume those buckets. And then … if you're someone who sits on the boundary between those things and you don't fit neatly into one of those, it's hard to have an organizational home, have a career path … [the] research software engineer designation … gives a name, and once you have a name you can … set up the career path and the organization so that you're not always trying to coerce yourself into being something you're not just so you can do your work and have your career.

Here, although the point is made in more career-oriented terms, the focus is still on RSE as a category that can help resolve professional identity problems at both personal and organizational levels.

Although RSE was seen as a positive professional identity by many, others had concerns that it implied a subordinate role that could be career-limiting. One interviewee noted that their organization had distinct research and non-research career paths, and that people with PhDs were less likely to identify as an RSE, regardless of their actual job description, for fear that it could limit their career opportunities on the research track. They noted that this was less of a concern for those with Bachelor's or Master's degrees who did not expect to be able to advance on the research track. (Note, however, that several interview subjects who described themselves as RSEs did have PhDs.) The inclusion of the term "software engineer" in the RSE title was a sticking point for some PhDs, leading them to distance themselves from the RSE role:

> A lot of times, you don't get your PhD in computer science to be called a software engineer. It conjures up that you write code, but may not understand much beyond the code … we need those people … [but] my coding is in support of a bigger picture.

Again, it is worth noting that several other interviewees saw precisely this aspect of their work – coding in support of a bigger picture – as a defining feature of the RSE role.

Multiplicity models

Interviewees who used multiplicity models to describe their professional identities typically did so to emphasize the fluidity of their professional role, or as a way of expressing support for the RSE community while identifying themselves as an outsider to that community. One variant of professional fluidity was interviewees who described themselves as partly an RSE and partly something else. This falls under the augmentation model because the individuals in question distinguished between different aspects of their role, but talked about these aspects as complementary rather than mutually exclusive. For example, one subject who actually had the job title of RSE and actively participated in the RSE community stated that they identified more as a "pure software engineer," noting at one point in the interview that "80% of what I do is probably pure software engineering, that other 20% goes on the side of what a research software engineer would do." At an earlier point in the interview, they had clarified the difference between the roles:

> the research software engineers probably … [work] more in the actual research side of things … I don't do very much of the research side of things, I do a lot of the infrastructure, of the testing, of you know the code quality.

The interviewee further characterized the "research side of things" as anything that involved directly modifying a research team's code, such as to improve performance, and the software engineering side as involving "creation of tools to help with the productivity of research scientists." Some other interviewees also made this distinction between software engineering being more infrastructure- and productivity-oriented and research software engineering being more about modifying code written by, or in cooperation with, research scientists.

Some interviewees noted a temporal aspect to their role fluidity, where they considered themselves to be moving back and forth between professional roles depending on the current

content of their work. This connects to the compartmentalization model of professional identity, where an individual switches between roles while maintaining a strong sense of boundaries between them:

> Before I started this new research project, I was doing very little research software engineering, and it's in part just because the project … had matured to the point where there wasn't as much research going on as there was software engineering. As we start up this [new] project, I'm probably gonna start off doing some computer science and then as we get further into the research of it, the PhD computer scientists are gonna take over most of the research, and I'll probably shift to a more research software engineering role of cleaning up and productizing that research.

A final type of multiplicity model was where interviewees expressed multiple identities, but defined a boundary around the RSE role and put themselves outside of it, even if they had a positive view of the role. As an example, one interviewee who had been involved in RSE professional activities defined themselves as "a software engineer that has a deep background in numerical and scientific computing," noting that "I do a lot of the work that could be done by an RSE but I don't consider myself an RSE because I have deep knowledge in numerical algorithms." Another interviewee with a PhD in computer science stated that "I wouldn't say that I'm an RSE per se" but that "I like being involved in the RSE movement because I think it's good to get those folks recognition." This person, like the interviewees above, drew a distinction between their work to develop software infrastructure and the RSE role, which they viewed as having more direct involvement with a scientific research team:

> I would say I am not like most of the people who are … in the RSE group … it seems like a lot of RSEs are kind of tacked onto a research group … I guess the place where I sort of differ [is] my background isn't necessarily like building scientific software … I build systems stuff, performance tools or systems software, I'm a systems person … we're actually doing the CS [computer science] research ourselves, so I would consider myself a CS researcher and a software developer but not necessarily an RSE.

As these examples illustrate, these interviewees were most concerned with distinguishing the RSE role from general software engineering and from computer science.

Overall, many of those who adopted multiplicity models of professional identity saw a defining characteristic of RSE work as engaging directly with a scientific research team or contributing to a research code. They considered more infrastructural aspects of software engineering – such as testing, managing productivity tools, or DevOps more generally – to be "pure" software engineering in the sense that it was not specific to the scientific computing context. On the computer science side, PhD computer scientists in particular were careful to distinguish aspects of their work which they considered computer science research from the more supporting role they viewed as typical of RSEs.

Summary

Overall, interviewees who followed dominance models of professional identity were more likely to see themselves as strongly attached to the RSE role, or to strongly reject the role. Those who were strongly attached to the role spoke about the role and the movement in more personal terms, as something that provided them with a sense of identity and belonging that they had previously struggled to find. In contrast, those who rejected the role seemed to feel it might be a problematic identity with negative impacts on their careers. In either case, these interviewees appeared to see their work role as more deeply connected to their sense of self than those who expressed multiple professional identities.

Interviewees who emphasized multiplicity of identities seemed to see research software engineering less as a personal and professional affiliation or identity, and more as job description which they sometimes fit into. They appeared less concerned about research software engineering providing them with a positive professional identity or feeling of belonging, and in some cases seemed to value not being tied to a specific role or career path. Many did, however, appear to see RSE as a useful category that might enhance the perceived value and importance of crucial aspects of their scientific software work.

Research software professionals at national laboratories

14 of the 17 research software professionals interviewed for this study worked at national laboratories, including Los Alamos National Laboratory, Lawrence Livermore National Laboratory, Sandia National Laboratories, and Oak Ridge National Laboratory. These laboratories each have distinct cultures around scientific computing, and the RSE role has taken on different meanings in each of them. The following narrative is based on interviews as well as the author's experience working with scientific computing teams at LANL and in the multi-laboratory Exascale Computing Program (ECP). Some elements are somewhat speculative and more research would be required to draw authoritative conclusions.

ORNL and SNL have dedicated research software engineering organizations, use RSE more widely as a job title, and deliberately seek to hire RSEs, in contrast to LANL and LLNL, which have been slower to embrace the RSE role at an institutional level. While the reasons for this are not entirely clear, it may have to do with ORNL and SNL having more diverse science and engineering project portfolios compared to the heavier physics emphasis of LANL and LLNL.

At LANL and LLNL, many members of the scientific computing community work on large multiphysics code projects funded by the National Nuclear Security Administration (NNSA). Perhaps because of their size and long-term stability, these projects have developed a highly differentiated set of roles around scientific computing. Although most of those involved in these projects are PhD scientists, the projects have historically valued software skills and appear to have provided relatively stable career paths for people who primarily focus on software development. They also enable scientists to focus their careers largely on software development while still being identified as scientists rather than software engineers. This

culture appears to have carried over, to some extent, to basic science projects at these laboratories that develop and rely on scientific computing codes. Where LANL and LLNL appear to differ from each other is primarily in the opportunities they provide for non-PhD software developers and engineers. While PhD scientists still clearly have higher status at these institutions, LLNL appears to more actively recruit and promote skilled software developers who do not have PhDs. This may in part be due to LLNL's proximity to a large pool of highly qualified software developers in the San Francisco Bay Area.

Overall, at LANL and LLNL, staff who do research software engineering tasks often have PhDs, and may be considered more as equal partners in the larger scientific enterprise in the sense that they can do software engineering work without having to renounce their identities as scientists. Perhaps because this arrangement already works relatively well, there may be less incentive for them to identify as RSEs, since in many cases this could imply a step down in the institutional status hierarchy, from a scientist role to an engineering role. SNL and ORNL, by contrast, may not make as strong status distinctions between scientists and engineers, or may have more experience with the idea that people can have career tracks that support scientific projects in an integral way without necessarily being scientists themselves. Interviewees at SNL and ORNL also seemed to have less status anxiety than those working in academic institutions, suggesting that the RSE role may be a more natural fit to how business was already done at these laboratories, rather than the more transformative professional role it implies for many working in academic settings.

At LANL, interviews also helped identify outliers in the laboratory's scientific computing community. These outliers were largely in organizations within the laboratory that do less computationally-intensive work and focus on smaller, shorter-term projects, some of which are more analysis-oriented than research focused. These projects often require computing support, but typically do not involve large, complex scientific codes. There appears to be a growing group of staff that specialize in computational support for research and analysis projects like these. These staff often have Bachelor's or Master's degrees and are possibly less recognized for their work than their counterparts elsewhere at LANL, or their peers at ORNL or SNL. They are also less aware of the RSE role and movement, even though they appear to have much in common with their counterparts in academia and could probably benefit from a stronger, more recognized professional identity.

Based on these considerations, it is possible to identify at least three distinct communities of people doing RSE-like work at the national laboratories:

1. Those working in dedicated RSE organizations (identified at SNL and ORNL). These individuals often work on a wide variety of research software projects, have a diverse mix of academic degree levels and backgrounds, have RSE as their job title, and get strong institutional support for their career development as RSEs.

2. Those working on large multiphysics or basic science code projects (identified at LANL and LLNL). These individuals appear to work almost exclusively within a high-performance computing context, are more likely to have PhDs, are closely integrated into large scientific projects, and work in a context where scientists can do RSE-like work while keeping their identities as scientists. While their expertise may be highly valued, they may have less of a sense of community or institutional support because they do not have dedicated professional organizations to support them.

3. Isolated research software professionals (identified at LANL). These are research software professionals who are often embedded within smaller research or analysis groups or projects, may work in organizations that place less value on computing skills, often have Bachelor's or Master's degrees, may have fewer opportunities for career advancement, and appear to be less aware of the RSE movement.

Further research would be necessary to identify the full scope of these communities and which laboratories and organizational units they can be found in. In addition, there may well be other distinct communities of research software professionals at the national laboratories that did not appear in this limited study.

Professionalization in software

The sociology of professions

The sociology of professions examines professions as social institutions, and seeks to explain how they emerge, evolve, and sometimes fade away over time. Historically, one influential approach was to enumerate traits or stages of development that distinguish professions from other occupations. This approach often focused on relatively powerful professions such as law and medicine, and identified characteristics such as strong professional associations, training and licensing requirements, codes of practice and ethics, and disciplinary mechanisms as key characteristics of professions. These institutional structures in turn enabled professions to restrict the number of professionals while also asserting exclusive control over a problem domain, thereby enhancing the status and pay of members. The general idea was that these institutional structures represented a kind of standard pathway for an emerging profession to follow in order to achieve maturity (Abbott 1988). This view of professions is now seen as overly simplistic by sociologists, but is still the model many would-be professions use as they seek to solidify their professional status.

Most recent work in the sociology of professions draws on Andrew Abbott's influential 1988 book *The System of Professions.* In this book, Abbott argues that professions should not be seen as isolated entities, but as part of a dynamic, evolving system in which professions engage in ongoing struggles to claim exclusive jurisdiction over particular problem domains. These

domains are rarely static because knowledge and technology change over time, opening up new areas of practice while other domains become obsolete. This leads to professions competing for control over new areas of practice, which can result in a continual churn of professions expanding, contracting, splitting, merging, or uneasily coexisting within a domain. In this view, claiming exclusive jurisdiction over a domain is the key characteristic of a profession – which often, but not always, includes the institutional structures described by earlier research on professions.

Abbott's model has also been subject to criticism. For example, Adams (2007) suggests that it overemphasizes conflict as the driving force in professional change when there is evidence that professions often resolve jurisdictional issues in cooperative ways. Adams also suggests that as professions work has proliferated, exclusive control over professional jurisdictions has become less valuable, meaning that "classic" strong professions like medicine and law may not be a good model for the future evolution of professions. More generally, the sociology of professions, as currently envisioned, is excessively focused on professional structures and underemphasizes less tangible elements such as professional identity, communities of practice, and the role of professions in stewarding new knowledge and skills.

History of the software engineering profession

The computing professions, specifically, have been cited as an area where both the classic model of institutional development of professions and the jurisdiction-focused approach of Abbott break down (Adams 2007). In particular, fields like programming, software development, computer science, and software engineering generally have poorly defined boundaries, few formal training or licensing requirements, and lower social status in some respects compared to professions like law and medicine. Large and well-developed professional societies do exist in computing, but have been somewhat less stable and exert less control over their members than professional organizations in classic mature professions. Nonetheless, it appears that the computing professions are flourishing, with high pay, many available jobs, active professional communities, and a high degree of professional autonomy for many practitioners. This situation has persisted for decades, suggesting that the ability to make strong, exclusive claims to a professional jurisdiction is not crucial to the success of these professions.

These characteristics are partly a result of how the computing professions have developed historically, the topic of Nathan Ensmenger's 2010 book *The Computer Boys Take Over: Computers, Programmers, and the Politics of Technical Expertise.* (The following account is derived from this book except where otherwise indicated, and focuses on the U.S. situation, which was not always replicated in other countries.) Ensmenger traces the roots of professional ambiguities in computing to the field's initial split between academic computer scientists and business programmers. In particular, businesses and business-oriented programmers often saw computer scientists as being overly concerned with theory and inattentive to the practical needs of business programmers, who were beginning to develop their own distinct approaches to the practical craft of programming. The lack of integration between these two sides of the field led to many disputes over the nature of computing as a profession, including whether

programmers should be considered subordinate technical workers similar to data processors or accountants, or higher-status scientifically trained experts; this in turn led to disputes over the need for a licensing process for practicing programmers, and whether a computer science education was a useful or even relevant qualification for programmers.

These disputes were reflected in a split between the main professional organizations in the field, the Association for Computing Machinery (ACM), which was initially largely an academic organization with few practicing programmers as members, and the Data Processing Management Association (DPMA), which aimed to represent business programmers alongside other information-oriented business workers. Most programmers, however, had little use for either organization, as their skills were in high demand, and the lack of standardization of programming knowledge allowed them to maintain an image as highly skilled practitioners of an esoteric art, which in turn allowed them greater autonomy and made them difficult to replace. Under these circumstances, they saw the push to standardize the programming profession as something that was more likely to benefit employers than themselves. As a result of these events, strong professional organizations failed to materialize and no consensus emerged about exactly what specific types of work computing professionals might claim exclusive jurisdiction over.

One turning point in the professional organization of computing was the emergence of software engineering as a professional field covering the practical aspects of software design and development. Although there are many origin stories for software engineering, most sources agree that it really took hold with the 1968 NATO Conference on Software Engineering. This conference was organized around the idea of a "software crisis," in which "large software development projects had acquired a reputation for being behind schedule, over budget, and bug ridden" (Ensmenger 2010, 196). This crisis was tied to the rapidly increasing scale and complexity of software projects and lack of clear professional or methodological standards in software development. Although, as Ensmenger notes, at this point it was "more an expression of ideals than a well-defined agenda," the concept of software engineering appealed to many different constituencies in the software community. For leaders of large software projects, it was appealing because it linked software practice to an established set of engineering fields with a proven record of successfully managing large projects, and suggested links to existing engineering methods that could be adapted to a software context. For software developers, it offered a kind of middle road between the more academic context of computer science and previous professionalization efforts that had envisioned them as "programmers" and tried to place them in a more subordinate organizational role. In addition, it provided an umbrella under which programming practice was more successfully integrated into professional organizations like the ACM and Institute of Electrical and Electronics Engineers (IEEE), and clarified the relationship between software practice and computer science. Overall, it provided a framework for the software community to start thinking about how to be more systematic in their practice while still allowing room for creativity and practical expertise.

Despite its appeal as a concept, software engineering has not come together as a profession in exactly the way its initial proponents might have hoped. One issue is that it has never been

quite clear how much software engineering really has in common with other engineering fields, or what types of engineering methods might be directly applicable to software. In addition, while software engineering has been integrated into professional societies like ACM and IEEE, it has never had strong institutional structures of its own, and as a field has never had mechanisms for enforcing training or licensing requirements similar to other areas of engineering. As such, it has functioned more as a discipline and set of methods than a coherent profession in the classic sense. As a result, the term software engineering is still used inconsistently as a professional category, with many practitioners and employers preferring terms like software developer or software architect, and with the title of software engineer being applied to people from a wide variety of educational backgrounds. Still, over time, software engineering has been successful in developing new subdisciplines, methods, and standards of practice and does appear increasingly coherent as a field.

This history raises the question of why software engineering has not followed the traditional professional model of trying to claim exclusive jurisdiction over a set of task areas. One possible reason is that software practice is so closely tied to specific computing technologies, which in turn have developed at a very rapid rate since their origins, making it difficult to nail down a single set of skills and methods that define the field. Also, as mentioned previously, it may simply be that the classic features of a profession are no longer particularly relevant to emerging fields of practice. But the main reason is probably the continued strong demand for software professionals, such that practitioners continue to be indifferent about the need for a strong professional organization to advance their interests. Given this position, it appears that many software professionals are more interested in sharing knowledge and building a strong professional community than in developing strict standards and defined boundaries for their field.

Research software engineering as a profession

The emergence of research software engineering as a professional field in relation to scientific software has arguably taken place in a very similar context to how software engineering emerged in relation to the wider field of computing. As with general software in the late 1960s, scientific computing is currently perceived as being somewhat in a state of crisis, with scientific code projects becoming increasingly large, complex, and difficult to manage. At the same time, scientific code developers are often not computing professionals, and have historically shown little interest in formal software development methods or standards. However, this is starting to change as scientists increasingly recognize code development and maintenance as key elements of their research productivity. Much like software developers in the 1960s, many research software developers recognize their increasingly important role but feel somewhat undervalued professionally, and see an opportunity to enhance and clarify their role by developing a new professional identity. However, the benefits of identifying or affiliating specifically as a research software engineer are not always clear to scientific software developers, including those interviewed for this study. Overall, scientific software developers are similar to other software developers since the 1970s in wanting to focus on professional community and knowledge sharing rather than on setting sharp boundaries around their field.

Conclusion

The RSE role is still relatively new and remains somewhat vaguely defined, both at an institutional level and for individuals who are trying to decide whether the RSE role is relevant to their work. While some individuals interviewed for this study strongly identified with RSE as their professional role, many others saw it as just one element of their larger role as scientific software professionals. These ambiguities appear typical of the early development of professions, particularly for recently emerging professions like software engineering, which often have looser boundaries than "classic" professions like law or medicine.

While the exact boundaries of the RSE role are still vague, interviews suggest that people who identify as RSEs see their role as closely related to software engineering in general, as well as management of RSEs, while computer scientists and HPC specialists often see themselves as occupying a distinct technical role from RSEs. In general, when interviewees did make strong distinctions between RSE and other scientific software roles, they often characterized RSE work as involving direct development or modification of scientific codes and close interaction with scientific research teams, distinguishing it from work that more indirectly supports the scientific enterprise, such as general DevOps support or developing software libraries that are widely used by scientists.

The different identity models and work situations of the software professionals interviewed for this study may have a significant impact on the way they engage with RSE as a professional role or movement, now and in the future. These differences may have practical implications for the RSE movement as it seeks to expand its reach within the scientific computing community.

Identity models may shape what kinds of engagements and types of communication software professionals would prefer to have with RSE organizations. People who adopt dominance models (strongly associating with a single professional identity) may not be particularly worried about the exact boundaries of the RSE role or profession, and might be more likely to seek out RSE organizations for a sense of personal validation and connection to a community of peers. These individuals may be more likely to enthusiastically buy into the RSE movement and perhaps commit time or take on leadership roles within organizations like US-RSE. People who adopt multiplicity models of professional identity appear to view RSE more as a set of activities they sometimes engage in, rather than a fixed professional identity, and seem to have narrower views of the RSE role. Their willingness to engage with the RSE movement might be highly sensitive to the perceived boundaries of the field, and they may be more interested in engaging in training and knowledge-sharing activities on an as-needed basis than in activities that draw them into a close community of peers. The leadership of RSE organizations may need to take these different orientations to the RSE identity into account to most effectively engage with and expand the RSE community.

In relation to national laboratories, the different work situations identified in this study may have implications for efforts by RSE movement organizations to recruit potential affiliates in these organizations. People who work in dedicated RSE organizations appear to be highly aware

of the RSE role and movement, not surprisingly, and are well-represented within the ranks and leadership of the US-RSE association. Little additional outreach effort is necessary to reach these individuals. Software professionals who work in multiphysics code development settings may pose more of an outreach challenge, although some of them are already involved with the US-RSE association. Finally, isolated research software professionals at the national laboratories appear to have very little, if any, representation in RSE organizations, and are a population that might respond well to outreach from these organizations.

These nuances in how people perceive the RSE role may create challenges for RSE professional organizations, like US-RSE, that promote a very broad definition of RSE that includes anyone involved in software work that contributes to science. Some potential questions include:

- Is RSE a term that can effectively cover the whole gamut of professionals working in scientific software, many of whom do not see themselves as engineers?

- If not, are there other scientific software roles that could be more clearly defined and are perhaps worthy of development as distinct professional areas or job titles in their own right?

- For RSE professional organizations, what is the right mix between serving the needs of:

    - Those who strongly identify as RSEs, who may be looking more for a sense of community and a clearly defined professional role, and

    - Those who view RSE more as a set of skills in their broader professional arsenal, and may want to participate in the RSE community more as a learning opportunity?

Although these are important questions that can help guide efforts to further professionalize research software engineering, the history of the broader software engineering profession suggests that there may be advantages in not prematurely settling on definitive answers. In the early history of software engineering, efforts to set sharp boundaries around the field or establish formal qualifications or credentials repeatedly failed because software developers saw these efforts as constraining rather than enabling their careers. By leaving the boundaries of the field relatively open, and focusing on development and sharing of new ideas, methods, and best practices, the software engineering community ensured the continuing relevance of the field and laid the groundwork for later efforts to develop and consolidate voluntary standards of practice. Continuing strong demand for software developers made this possible. As long as demand for people with RSE skills continues to grow in scientific workplaces – which seems highly likely – maintaining these ambiguities as active topics of discussion while supporting an open and vital community is likely to be the most productive path for ensuring the future of the RSE profession.

References

Abbott, Andrew (1988). *The System of Professions: An Essay on the Division of Expert Labor.* Chicago: University of Chicago Press.

Adams, Tracey L. (2007). Interprofessional relations and the emergence of a new profession: Software engineering in the United States, United Kingdom, and Canada. *The Sociological Quarterly* 48: 507-532.

Caza, Brianna Barker and Creary, Stephanie (2016). The construction of professional identity. In Wilkinson, Adrian, Hislop, Donald and Coupland, Christine (eds), *Perspectives on Contemporary Professional Work: Challenges and Experiences.* Cheltenham: Edward Elgar Publishing.

Ensmenger, Nathan (2010). *The Computer Boys Take Over: Computers, Programmers, and the Politics of Technical Expertise.* Cambridge, MA: MIT Press.